Journal of Building Engineering 47 (2022) 103822

Contents lists available at ScienceDirect

Journal of Building Engineering

journal homepage: www.elsevier.com/locate/jobe

Architectural layout design through deep learning and agent-based modeling: A hybrid approach

Morteza Rahbar [a], Mohammadjavad Mahdavinejad [b,*], Amir H.D. Markazi [c], Mohammadreza Bemanian [b]

[a] School of Architecture and Environmental Design, Iran University of Science and Technology, Tehran, Iran
[b] Department of Architecture, Tarbiat Modares University, Tehran, Iran
[c] School of Mechanical Engineering, Iran University of Science and Technology, Tehran, Iran

ARTICLE INFO

Keywords:
Spatial layout design
Space planning problem
Deep learning
GAN
Agent-based modeling

ABSTRACT

This paper presents a novel hybrid approach for generating automated 2D architectural layouts by combining agent-based modeling with deep learning algorithms. The primary goal of this research is to maintain the designers' high-level, supervisory control over the generated results and process, allowing them to manage the whole process so that the created results satisfy the desired topological and geometrical constraints. The proposed hybrid approach consists of two different methods. First, hierarchical phases of agent-based modeling are simulated to generate a bubble diagram that satisfies the topological conditions. A rule-based algorithm converts bubble diagrams into heat maps. Second, the pix2pix algorithm translates the heat maps into an architectural spatial layout as a conditional GAN and deep learning approach. In doing so, a unique dataset was manually generated, and the cGAN algorithm was trained based on this dataset. The hybrid method of these processes makes it possible to generate an architectural layout based on a particular footprint and desired high-level constraints. The findings of agent-based modeling showed complete consistency with the required topological requirements, whereas deep learning results demonstrated the ability of cGAN to satisfy geometrical constraints learned throughout the training phase. The hybrid method's results showed enhanced computational accuracy in generating synthetic architectural layouts compared to previous studies.

1. Introduction

The preliminary proposition of space layout is an essential starting phase of any architectural design process. They are designed in accordance with high-level topological constraints determining the geometrical features of each space. In case a designer encounters many difficulties concerning complex topological relationships, the generative architectural layout techniques may assist in finding a new approach to solve the problem. These techniques are taken as a part of the computer-aided design concept, and they never aim to replace human designers. Such methods could facilitate the architect's decision making in the early stages of a design process. In the last few decades, many scholars studied generating automated layouts for architectural designs. Their research yielded promising results (see section 2); however, conditioning the generated layout to some input constraints such as the location of the entrance door, windows, and available walls (or structure), as well as conditioning the layout to the desired topology, could generate a more practical layout. This paper concentrates on generating automated 2D layouts conditioned to a fixed footprint, the entrance door, the windows

* Corresponding author.
E-mail address: mahdavinejad@modares.ac.ir (M. Mahdavinejad).

https://doi.org/10.1016/j.jobe.2021.103822
Received 19 August 2021; Received in revised form 16 November 2021; Accepted 3 December 2021
Available online 15 December 2021

and a desired topological constraint, all of which are specified by the designer as inputs to the proposed algorithm.

Sample high-level conditions and the suggested layout for an apartment are shown in Fig. 1. The left and the center images are the algorithm's inputs. The right image is the designed layout regarding the apartment's footprint (the entrance door, windows and walls) and the desired topology. The layout was based on the high-level constraints representation, a topological graph of space connectivity within a fixed footprint. Both topological and geometrical constraints were addressed in the space layout design. Geometrical constraints include the size of the rooms, their area and proportion. Topological constraints include the hierarchical arrangement of the space. Both geometrical and topological constraints result from the objective and subjective parameters.

Space layouts are classified as NP-complete problems, which means that they cannot be solved with definite optimality within a reasonable amount of time [1,2]. Space layout is identified as a multifaceted problem due to several reasons, including complicated relation of space layout parameters, difficulties in quantifying quality measures, and the combination of continuous and discontinuous exploration space [3]. Since the 1960s, one of the main streams of computer scientists' research has been the space layout design problem. Most of these studies focused on solving two key problems: the variant layout generative algorithm and the cost function algorithm to evaluate each generated layout. From an architect's point of view, both subjective and objective considerations are taken into account for judging a successful layout as a potential solution. In this case, there are usually numerous possible solutions underscoring the inherent uncertainty of the space allocation problem (SAP), which could not necessarily be modeled as a deterministic mathematical relation of spaces. To date, as a result of several SAP research works, the complex cost function has been simplified to a specific objective factor, which has been solved with an optimizer [3–12].

In this paper, a novel hybrid method of integrating agent-based modeling (ABM) and deep learning was applied to generate automated space layouts. In many design cases, typically, there are boundary restrictions, and the designers have to assign spaces inside the fixed footprint. Therefore, this research dealt with generating space layouts inside a fixed footprint. As the first step, the algorithm started with a pre-defined footprint on which the entrance, windows, and walls were all labelled. According to this step, the designer also specified the high-level constraints, illustrated as spaces topological connectivity graph. In the next step, several physical forces and agent-based modeling techniques were applied to generate a bubble diagram meeting the topological constraints. Then the bubble diagram was visualized as a heat map; each of its colors indicated the probability of space allocation in different parts of the exploration space. As the last step, a data-driven deep learning technique was utilized to convert the heat map to a labelled layout satisfying the geometrical constraints. The geometrical constraints were implicitly trained by a conditional generative adversarial network [13] with a specific dataset. Since the innovation of generative adversarial networks (GAN) with Ian Goodfellow's influential paper [14] in 2014, different branches of GANs have emerged. Each of these branches can be applied to deal with a specific task. Conditional generative adversarial networks (cGAN) are considered as one of the main streams [15], which can be used to generate a synthetic image based on an input image. In this research, a specific dataset was used to train cGAN so that it could generate an image of the space layout based on the heat map that was previously generated by agent-based modeling. The main contributions of this research are as follows:

- Generate automated space layout design through a novel hybrid method of ABM and cGAN
- Generate the bubble diagram and space allocation heat map through the agent-based modeling method satisfying the topological constraints
- Generate synthetic space layout through trained cGAN satisfying the geometrical constraints
- Prepare a new dataset which was used for cGAN training in automated space layout generation

The main objective of this paper is to propose a practical method for generating an automated synthetic layout that would allow the

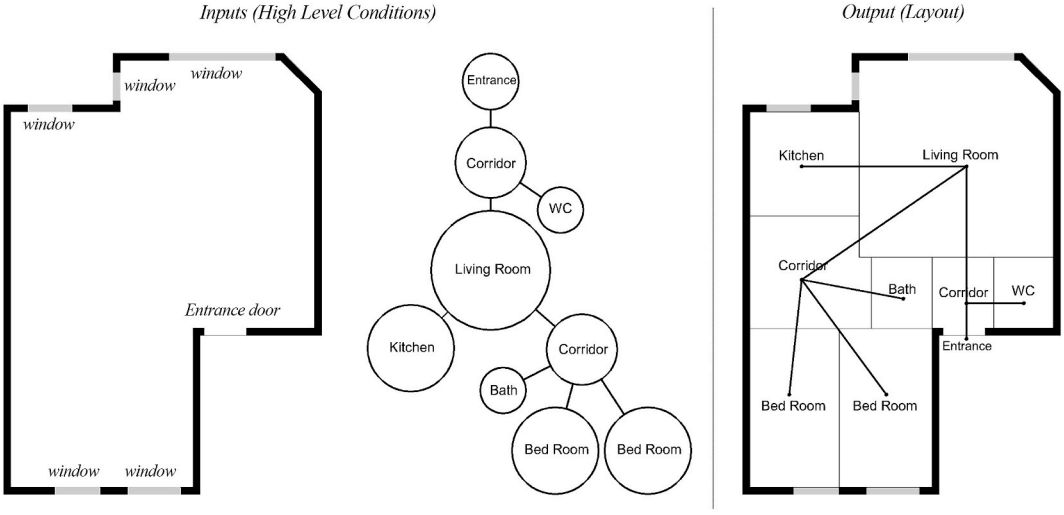

Fig. 1. An apartment footprint (Left), Topological connectivity graph of spaces (Center), A sample layout design (Right).

designer to have more control over the generated result. The designer may easily request the required topology in a defined footprint by conditioning this hybrid algorithm with high-level constraints. The background, related studies and research gaps are all thoroughly covered in section 2. In section 3, the research approach is mentioned in detail. The agent-based modeling technique used in this study and the cGAN model are described in detail in section 4. The results and discussions are reported in section 5 and section 6 concludes and suggests future works. In Table 1, the abbreviations used in this study are listed.

2. Background

A space layout design is defined as a creative activity through which several possible solutions could be obtained. The creativity in a design task, therefore, has an intrinsic uncertainty, making it difficult to be formulated as a specific objective function [2,16]. Design problems are generally non-operational, which distinguishes them from the majority of scientific research topics [17,18]. The space layout design problem has been introduced as one of the main challenges of scholars over the last few decades. Space planning and architectural design optimization are the other general terms applied for this problem, and defining the allocation of the spaces based on topological and geometrical constraints is considered their common goal. There are also other general forms of layout design problems, including assignment problems or bin packing problems. Researchers have put forward several methods to tackle these problems. For instance, the greedy algorithm, dynamic programming, and mixed-integer programming have already been proposed [2]. However, the layout design problem differs in terms of the spatial constraints which must be considered. The space layout problem falls into the category of NP-complete problems, which entails various requirements to meet the topological and geometrical constraints.

Numerous methods have been proposed to solve the space layout design problem. The very first stages of interest in the field appeared in the early 1960s. Levin [4], wrote a book on finding the optimum architectural layout using graphs. Since the early stages, graphs have been one of the key streams of architectural layout design. Grason investigated the graph as a new representation of floor plans [5] and explored how to apply it in computerized space planning [6]. Roth et al. [19] devised a method for turning graphs into rectangular floor plans. According to this research, three transforming steps from a sample graph to an adjacency matrix and finally, the transformation to a rectangular plan on a modular cell grid have been described. To solve architectural problems, Roth [20] also investigated the general application of graph algorithms. In recent years, some researchers have studied the graph algorithms, and adjacency matrix as a method for automated floor plan generation [21–23]. Wang et al. [24] worked on customizing and generating floor plans based on graph transformations. They used a dual graph to represent existing floor plans and applied two possible transformation rules to develop a new dual graph and subsequently generate and customize the new floor plan.

Apart from graph algorithm methods, other problem-solving approaches have also been taken into account. Generally, all of these methods share a common characteristic, that is, they all include two separate parts. In the first part, they attempt to define a function generating the synthetic layouts. In the second part, a functional algorithm is defined to evaluate the quality of the generated layouts based on topological and geometrical constraints. In some of these methods, a well-defined problem-solving technique was applied, while in others, an ill-defined problem-solving approach was used. The ill-defined problems are those which do not offer clear goals, solution paths, or expected solutions. The well-defined problems propose specific goals, clear-cut solution paths, and precise solutions [25]. The ill-structured problems are easily defined, and they do not require extensive problem knowledge. In this case, iterative methods are applied to obtain good results. On the contrary, the well-structured problems are hard to be defined, and direct methods are used to obtain the excellent quality of generated solutions [18].

Table 1
Nomenclature.

Abbreviation	Description
SAP	Space Allocation Problem
fp	The footprint of an apartment
ent	The apartment's entrance door
Z_{public}	The public zone of the apartment where the kitchen and living rooms are located
$Z_{private}$	The private zone of the apartment where bedrooms, bathroom and in-between corridors are located
S_i	The spaces of the apartment such as bedrooms, kitchen, living room, WC, etc.
W_i	The location of the existing windows (or possibilities of allocating the windows)
A	Adjacency Matrix
$area_i$	The area of space i
n_s	Number of spaces
L	Generated layout
Model	Trained cGAN model based on the prepared dataset
hL	Hierarchy levels
lev_i	The i th hierarchical level
S_i^x	The local x coordinate of space i inside the outline
S_i^y	The local y coordinate of space i inside the outline
C	The connection between spaces and a window
T_1	Direct Topological connections between spaces
T_2	Indirect Topological connection between spaces
Z_i	The i th zone
sZ_i	The i th sub-zone

Some scholars have investigated well-defined problem-solving methods concerning space layout design. Flemming et al. [26] applied a logic programming method for architectural layout generative design. They tried to consider a broad spectrum of criteria or concerns simultaneously. Duarte [27] applied discursive grammar in the design and production of mass customized houses. The discursive grammar employed in his research was a set of substitution rules applied reclusively to an initial assertion to produce a final statement. Before that, Stiny & Mitchell [28] applied parametric shape grammar rules of Palladio's villas to generate floor plans of Palladian style. Constraint-based simulation is considered as another well-defined method. In this method, a direct search approach is applied, which searches for the solutions through the domains according to the problem-specific mathematical model [29].

On the other hand, some other scholars have investigated ill-defined problem-solving methods for space layout design. Koeing & Schneider [18] described the capabilities of these methods in comparison to well-defined methods in space layout design problems. Approaches such as agent-based systems and evolutionary algorithms are examples of ill-defined methods requiring little problem knowledge. Guo & Li [30] applied the agent-based system for the topological solution and an evolutionary method for geometrical constraints. Spaces were represented as agents in their research, interacting with one another due to attraction and repulsion forces, and the spaces eventually stabilized in an equilibrium phase. The forces were defined considering topological constraints, and they could generate any form independent of any existing building footprint. Lastly, they applied an evolutionary method to generate the geometry of each space in the 3d cellular grid. Liggett & Mitchell [7] used quadratic assignment optimization with a single cost function to maximize workflow efficiency between activities. Since the 1990s, many researchers have concentrated on applying evolutionary algorithms [11,31] to solve space planning problems. Specifically, genetic algorithm [8,10,12,32] and genetic

Table 2
Literature summary table (in chronological order).

Ref	*Authors*	*Date*	*Method*	*Title*
[4]	*Levin*	*1964*	*Graph Theory*	*Use of graphs to decide the optimum layout of buildings*
[5]	*Grason*	*1970*	*Graph Theory*	*A dual linear graph representation for space-filling location problems of the floor plan type*
[6]	*Grason*	*1971*	*Graph Theory*	*An approach to computerized space planning using graph theory*
[28]	*Stiny & Mitchell*	*1978*	*Shape Grammar*	*The palladian grammar*
[7]	*Liggett & Mitchell*	*1981*	*Single stage process for Quadratic Assignment Problem*	*Optimal space planning in practice*
[19]	*Roth* et al.	*1982*	*Graph Theory*	*Turning a graph into a rectangular floor plan*
[26]	*Flemming* et al.	*1988*	*Expert System*	*A generative expert system for the design of building layouts*
[20]	*Roth & Hashimshony*	*1988*	*Graph Theory*	*Algorithms in graph theory and their use for solving problems in architectural design*
[8]	*Gero* et al.	*1997*	*Genetic Engineering*	*Learning and re-using information in space layout planning problems using genetic engineering*
[9]	*Jagielski & Gero*	*1997*	*Genetic Programming*	*A genetic programming approach to the space layout planning problem*
[3]	*Jo & Gero*	*1998*	*Genetic Algorithm*	*Space layout planning using an evolutionary approach*
[10]	*Gero & Kazakov*	*1998*	*Genetic Algorithm*	*Evolving design genes in space layout planning problems*
[29]	*Li* et al.	*2000*	*Non-Linear Programming*	*A constraint based generative system for floor layouts*
[11]	*Michalek* et al.	*2002*	*Mathematical Optimization + Subjective Decision Making*	*Architectural layout design optimization*
[31]	*Bonnaire & Riff*	*2002*	*Evolutionary Algorithms*	*A self-adaptable distributed evolutionary algorithm to tackle space planning problems*
[12]	*Bausys & Pankrasovaite*	*2005*	*Genetic Algorithm*	*Optimization of architectural layout by the improved genetic algorithm*
[27]	*Duarte*	*2005*	*Discursive Grammar*	*A discursive grammar for customizing mass housing: the case of siza's houses at malagueira*
[33]	*Inoue & Takagi*	*2008*	*Cellular Automata + Evolutionary Multi-objective Optimization*	*Layout algorithm for an ec-based room layout planning support system*
[32]	*Wong & Chan*	*2009*	*Graph Theory + Evolutionary Algorithm*	*Evoarch: An evolutionary algorithm for architectural layout design*
[34]	*Inoue & Takagi*	*2009*	*Evolutionary Multi-objective Optimization*	*Emo-based architectural room floor planning*
[18]	*Koenig & Schneider*	*2012*	*Evolutionary Algorithm + Hierarchical Structuring*	*Hierarchical structuring of layout problems in an interactive evolutionary layout system*
[35]	*Rodrigues* et al.	*2013*	*Evolutionary Strategy + Stochastic Hill Climbing*	*An approach to the multi-level space allocation problem in architecture using a hybrid evolutionary technique*
[1]	*Campbell* et al.	*2014*	*Dense Packing + Subdivision Algorithm*	*Comparing two evolutionary algorithm based methods for layout generation: Dense packing versus subdivision*
[21]	*Shekhawat*	*2015*	*Space Partitioning*	*Automated space allocation using mathematical techniques*
[36]	*Wortmann* et al.	*2015*	*Surrogate based Optimization*	*Advantages of surrogate models for architectural design optimization*
[2]	*Dino*	*2016*	*Evolutionary Algorithm*	*An evolutionary approach for 3d architectural space layout design exploration*
[30]	*Guo & Li*	*2017*	*Evolutionary Algorithm + Agent-based Modeling*	*Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system*
[22]	*Shekhawat*	*2018*	*Graph Theory + Mathematical Modeling*	*Enumerating generic rectangular floor plans*
[23]	*Slusarczyk*	*2018*	*Graph Theory*	*Graph-based representation of design properties in creating building floorplans*
[24]	*Wang* et al.	*2018*	*Graph Theory*	*Customization and generation of floor plans based on graph transformations*
[38]	*Huang & Zheng*	*2018*	*Deep Learning*	*Architectural drawings recognition and generation through machine learning*
[37]	*Rahbar* et al.	*2019*	*Deep Learning*	*Generating synthetic space allocation probability layouts based on trained conditional-GANs*
[39]	*Chailou*	*2020*	*Deep Learning*	*Archigan: Artificial intelligence x architecture*

programming [8,9] received significant attention. In recent years, some studies have concentrated on multi-objective optimization [33,34] and hybrid optimization techniques [35]. Wortmann et al. [36] applied surrogate-based models for architectural design optimization tasks, and they expressed the advantages of their method over genetic metaheuristic.

Regarding the methods mentioned above, it can be inferred that all approaches can be considered a rule-based method that attempts to simulate space layout design through some high-level rules. Either an ill-defined or a well-defined problem-solving technique could be applied in these rule design procedures. In 2014, Ian Goodfellow presented a novel method named GAN (Generative Adversarial Networks) for generating synthetic data. After that, many scholars worked on GANs and presented new branches of GANs capable of doing specific tasks. Generally, GANs are one of the mainstreams of deep learning generative algorithms. In the last few years, some scholars focused on applying GANs in generative architectural designs. Rahbar et al. [37] applied the pix2pix algorithm as a conditional GAN model for translating the image of an apartment footprint directly to a labelled layout design. Huang & Zheng [38] applied the pix2pix algorithm for two tasks; first for recognizing spaces in a layout and second for generating a furnished plan from a

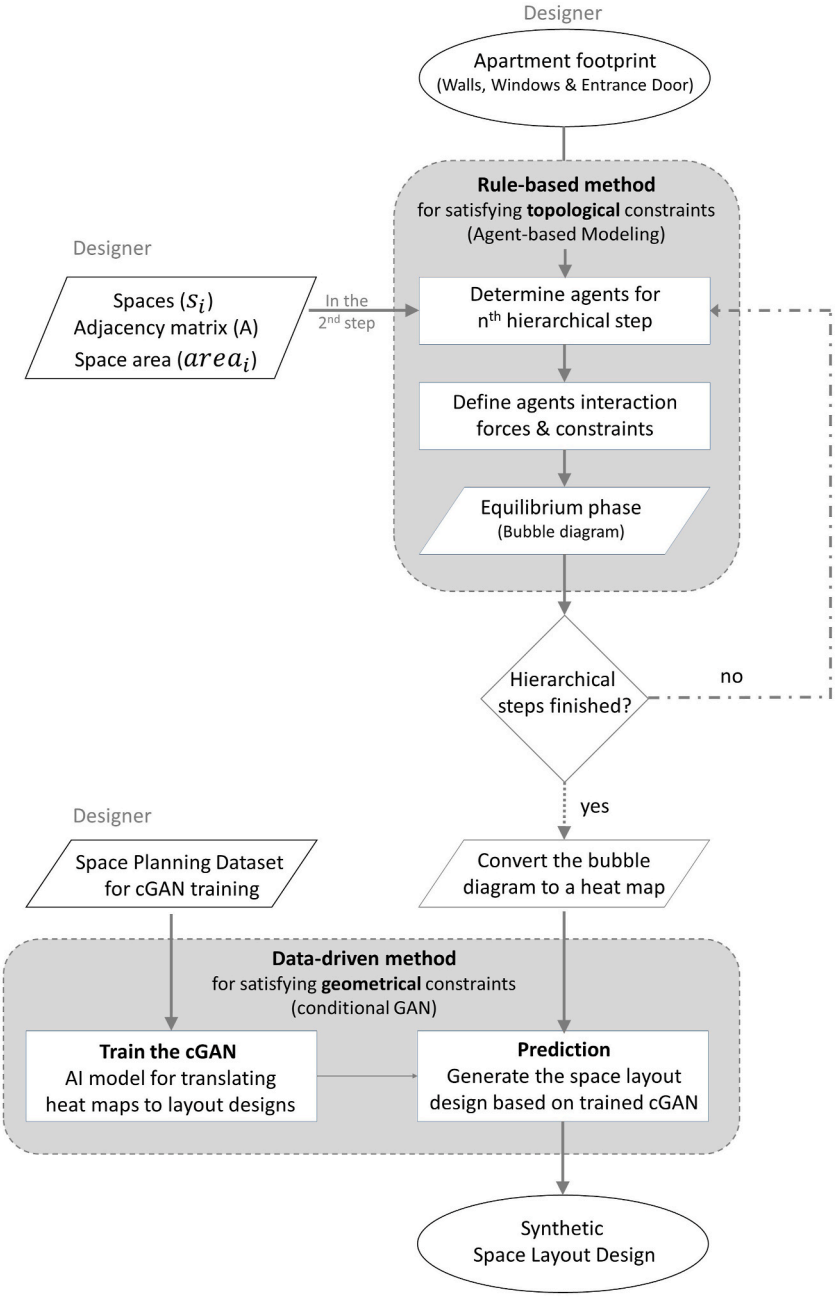

Fig. 2. Proposed automated design methodolgy

labelled image. Chailou [39] utilized pix2pix algorithm in three main stages. At the first stage, a building footprint was generated from a parcel outline. In the second stage (program repartition), each space was labelled with a color inside the footprint. In the third stage, the furnished layout was generated. These studies show promising results for generating building floor plans; however, a main problem still exists; the designer has no role in conditioning the result to the desired topological constraints. Pix2pix algorithm trains on pairs of images. The algorithm learns the procedure of translating an input image to an output image. It imitates the underlying conversion patterns in the training dataset's input and output images. In the previous studies, the generated topology of spaces was arranged completely by the pix2pix algorithm based on the shown dataset to the algorithm, which could be far from the designer's desired topology. These generated layouts are highly dependent on the quality and quantity of the dataset. Generally, more training datasets with approved layout designs may lead to better-generated layouts. Nevertheless, preparing a big dataset of layouts with similar topological concepts is a complex and time-consuming task. Even preparing such a dataset could result in a good case-specific trained model that generates reasonable layouts similar to the training dataset but cannot generalize to other layout designs.

To overcome this problem, this paper explores a novel approach based on a combination of rule-based and data-driven design processes. In this approach, the designer defines the desired topological relationship of spaces besides giving the base footprint. In most design cases, the architect prefers to apply a specific high-level topological constraint to the problem. In the first part of this research, agent-based modeling has been applied as a rule-based design method simulating the topological relations of spaces. Agent-based modeling is a heuristic ill-defined problem-solving technique that has been used here to simulate the location of spaces in any given building footprint. This agent-based modelling resulted in a bubble diagram and a heat map of layout design without considering any geometrical constraints. In the second part of this research, a trained pix2pix deep learning model has been applied to extract geometrical data from the existing layout designs. Any architectural layout designed by an architect is assumed to include hidden semantics and meaningful data. Thus, instead of applying a rule-based approach for the whole procedure of layout design, this paper studied the capabilities of data-driven methods for simulating the geometrical aspects of each space. The idea was to determine the dimension of each space, their proportion and area and any other geometrical features through the existing layouts designed by human intelligence (architects). In doing so, a specific dataset was prepared to train the AI model. The result of this method was a synthetic colored layout indicating the location and the geometry of each space inside the given building footprint regarding the topological constraints and the learnt geometrical aspects. This final spatial layout could provide some basic conceptual layout suggestions in the early stages of a design. The following section presents a detailed description of the research approach. The literature summary is presented in Table 2.

3. Research approach

The proposed algorithm entailed two different methods. The first method applied an agent-based modeling algorithm that was considered a rule-based method to generate a bubble diagram based on the given high-level features. Then the bubble diagram was converted to a heat map. The second method applied the conditional GAN algorithm as a data-driven model to generate the space layout based on the heat-map and the given dataset. Fig. 2 depicts the research methodology flowchart. As the first step, the designer specified the existing building footprint consisting of walls, windows and entrance door. The footprint was an influential high-level constraint which the designer aimed to allocate spaces (activities) inside this footprint concerning the entrance door and the windows. To generate a bubble diagram, other input parameters such as the spaces (s_i), their area ($area_i$) and the adjacency matrix (A) also were identified by the designer. These are the other high-level constraints that are generally known as the problem's independent variables. In this research, agent-based modeling included three sequential steps. The first step was to define the agents. In the second step, the agent's interaction forces and constraints were defined and simulated. The third step was the equilibrium phase, where the agents reached a stable position with no more effective resultant force. In this phase, the algorithm generated the bubble diagram indicating the topological relationship of spaces.

A hierarchical approach was used to create an accurate bubble diagram. The approach was based on the idea of partitioning the exploration space into several zones in consecutive loops until final sub-zones were generated. Ultimately, each activity or space was randomly distributed inside the boundary of its related sub-zone. This distribution was the initial position of agents before starting the simulation of interacting forces (Fig. 3). The interacting forces were defined regarding high-level topological constraints. Compared

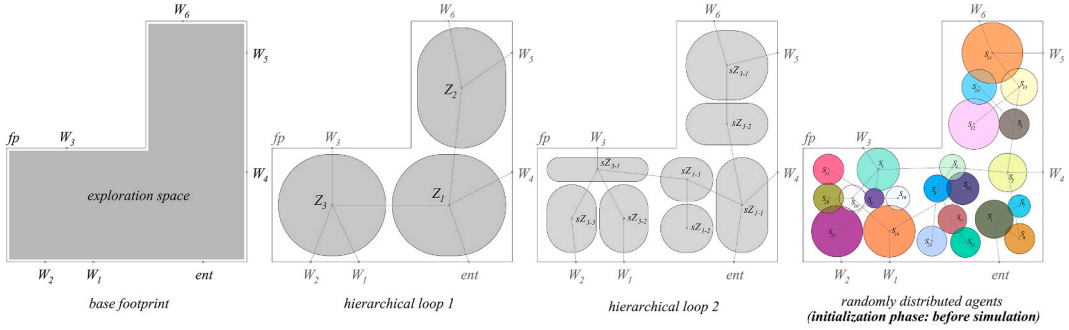

Fig. 3. The concept of hierarchical approach.

with single-step agent-based modeling, they were randomly distributed in their sub-zones instead of randomly spreading the agents all over the footprint. This approach reduces the computation complexity and ensures reaching a reliable and robust equilibrium phase.

The number of hierarchical loops is a hyperparameter that should be fine-tuned based on a project's scale and complexity. In this study, two hierarchical loops were used: the first loop partitioned the exploration space of an apartment footprint into private and public zones, and in the second loop, the agents were randomly distributed in their zones. After that, the agents' interactions were simulated until they reached an equilibrium state. This equilibrium state was the final bubble diagram that proposed each space's location in the footprint based on the topological constraints. The bubble diagram was then converted to a heat map, which was then given into the cGAN algorithm as an image. Pix2pix, as a cGAN algorithm, was chosen for this study. Pix2pix is an algorithm with the capability of translating an image into another image. The architecture of this algorithm is described in detail in section 4. The designer also prepared a specific space layout training dataset for the cGAN algorithm. This dataset necessarily included a pair of images indicating the before and the after image translation. The before images were the space layout heat maps, and the after images were geometrical space layouts. After training the cGAN algorithm with these pairs of images, the deep learning model could generate synthetic space layouts.

Fig. 4 illustrates the research procedure diagram. The diagram entailed four main parts. The first part was related to the high-level constraints specified by the designer. The adjacency matrix was the first input corresponding to a layout graph. There was also a predefined apartment footprint with windows and entrance door labelled. This footprint was the outline boundary for generating the

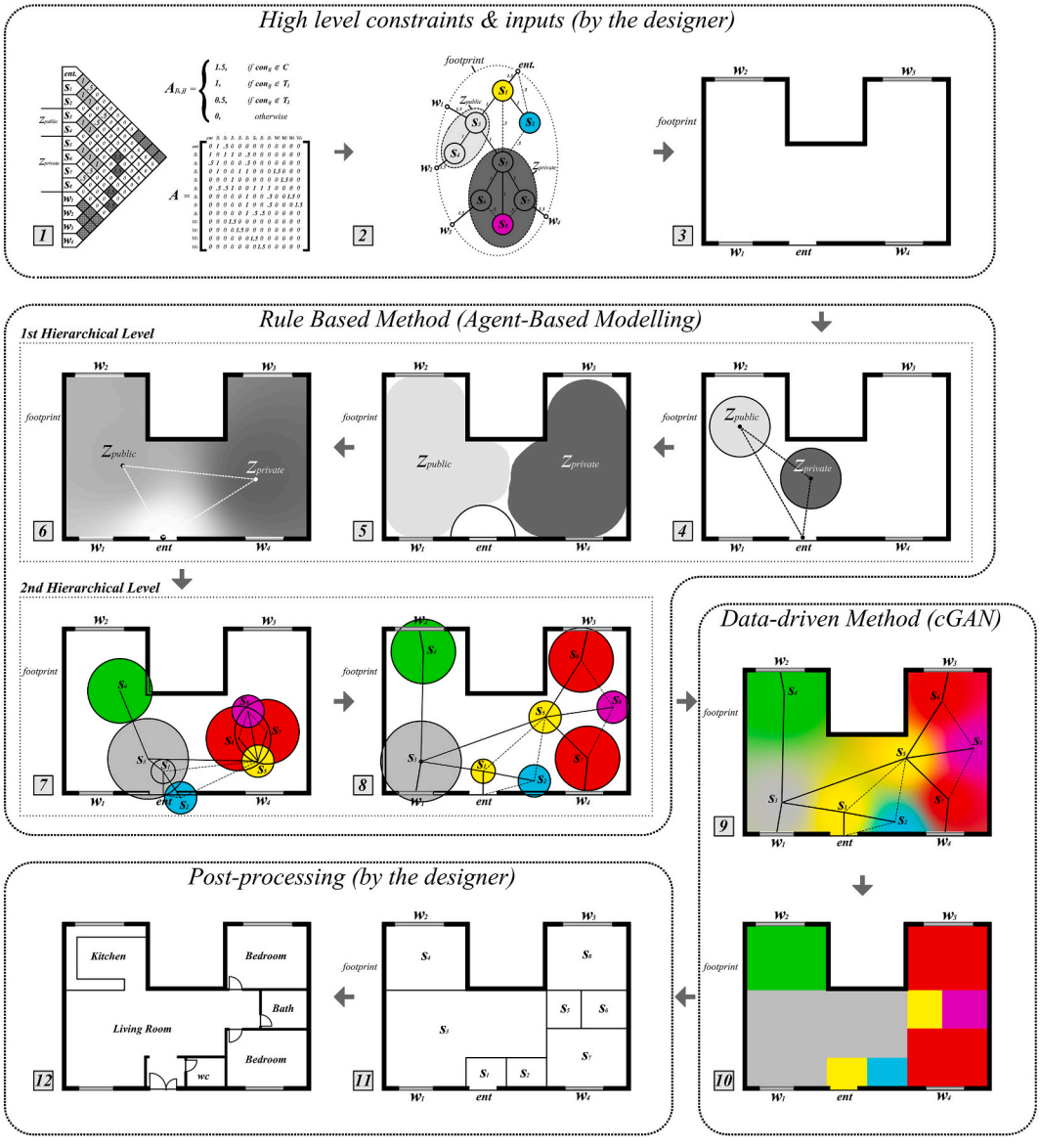

Fig. 4. Research methodology procedure (1-Highlevel constraint, 2-Agent-based modeling, 3-cGAN, 4-Post-processing).

layout design. A rule-based method was applied to find the topological graph based on high-level constraints in the next part. Specifically, the agent-based modeling technique was used. In this research, these agents were determined in two hierarchical loops, facilitating the simulation of topological constraints. The first loop aimed to simulate the topological relation of private and public zones. The second loop aimed to find the topological relation of spaces inside each of these zones. The number of the zones and hierarchical steps depended on the design project. In the next part, a data-driven method was applied to find a solution for the geometrical constraints. Conditional GAN was trained according to a specific dataset to translate the heat maps to layout designs. In the final part, post-processing steps were applied to transform the layout design to a conventional floor plan.

3.1. Hierarchical agent-based modeling

This research uses agent-based modelling as a rule-based design method to generate a bubble-diagram layout based on high-level constraints. Here, each agent represented a specific functional space and the forces were defined between the agents and other objects such as walls, windows and doors. These force vectors led the agents to move in the exploration area and interact with each other until they reached a general equilibrium phase in which there was no efficient force to change their position. Hierarchical agent-based modeling applies a similar concept, but it runs in several sequential steps. In the hierarchical approach, instead of defining each space as a specific functional space and finding the equilibrium phase in only one single run, a cluster of spaces is determined as zone agents, and in the final step, the functional spaces are represented as agents. Each hierarchical step aims to segment the overall space into smaller clusters. Each cluster consists of several spaces that are modeled as the new agents in the subsequent hierarchical step. Hierarchical agent-based modeling segments the exploration space into zones and the zones into sub-zones and repeats this loop until the final phase. Following this, each agent's initial position is randomly sampled in their respective sub-zones. In comparison to a single run of agent-based simulation, this technique ensures better initialization and results.

The Kangaroo add-on, Grasshopper-3D plugin and Rhinoceros software were used for simulating the interacting forces. In this research, two hierarchical steps were simulated to gain good results for apartment space planning. The number of steps is an experimental hyper-parameter that highly correlates with the complexity of the problem. In the first step, the apartment was divided into two clusters of public zone denoted as Z_{public} and private zone denoted as $Z_{private}$. Kitchens and living rooms were regarded a subset of the public zone, while bedrooms, bathrooms, and other private spaces were considered a subset of the private zone. At first, these two zones were defined as circles positioned randomly inside the footprint. Each of these circles included one hundred points equally spaced on the perimeter of the circle. These points were the agents interacting with each other, and after the equilibrium phase, they defined the public and private zones of the layout (Figs. 5 and 6).

Several experiments have been performed to simulate the planning of the private and public zones to obtain the best combination of force vectors. Finally, the best experiment indicated that defining four force vectors could generate an accurate layout design for the private and public zones: 1) $F^i_{boundary}$ for keeping the agents inside the footprint, 2) F^i_{area} for restricting the area of the zones to the

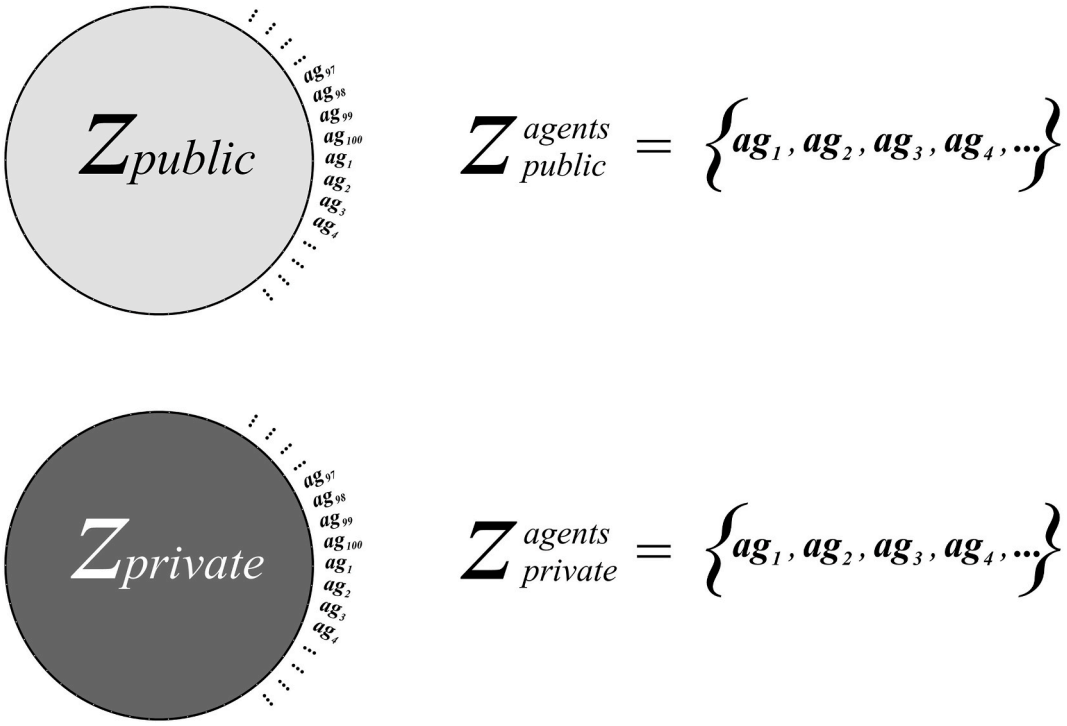

Fig. 5. Agents defined for the public and private zones.

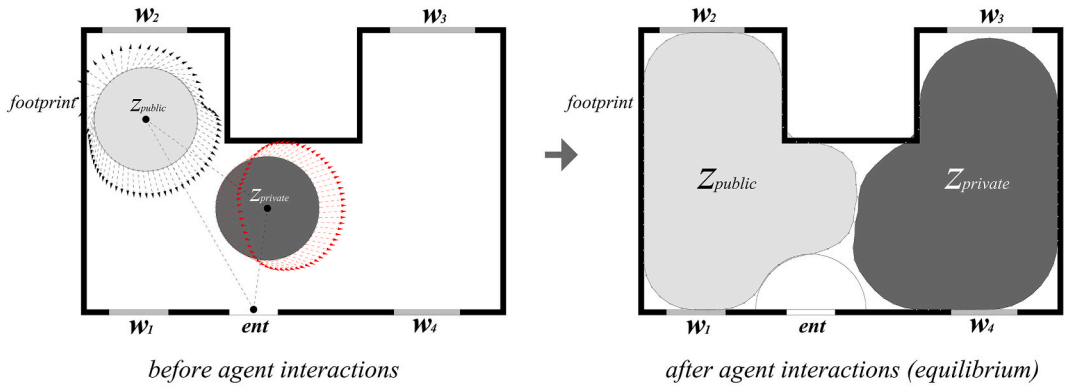

Fig. 6. Equilibrium phase of the first hierarchical step.

predefined values, 3) $F^i_{collision}$ for avoiding the zones from colliding to each other and 4) F^i_{spring} for defining spring forces between adjacent agents on the perimeter of the zone boundary. The resultant of these vectors F^i_{Total}, forced the agents to move in the exploration space (Equation (1)). After several iterations, each agent reached a specific location where there was no more efficient resultant force, and they reached their equilibrium position. The form of the original circle varied as the agents moved through the iterations, ensuring free form-finding. Fig. 7 illustrates the resultant force vector F^i_{Total} and its components.

$$F^i_{Total} = F^i_{spring} + F^i_{area} + F^i_{collision} + F^i_{boundary} \qquad i \in \{s^{agents}_{public} \cup s^{agents}_{private}\} \tag{1}$$

The spring force was designed to maintain the consistency of the zone's shape. Each agent had two spring forces, either attracting or repulsing the two adjacent agents. The resultant of these two forces is denoted as F^i_{spring}. The rest length is a hyperparameter depending on the size of the boundary. The agents were forced to merge into a single agent when the rest lengths were set to small values, while the zone's borderline was stretched when the rest lengths were set to large values. However, the area force F^i_{area} played a significant role in keeping the form at a specific predefined area. Whenever a part of the zone boundary crossed the footprint, the agents received a backward force called $F^i_{boundary}$, keeping them inside the exploration space. In case of zone collision, the collided agents received a force

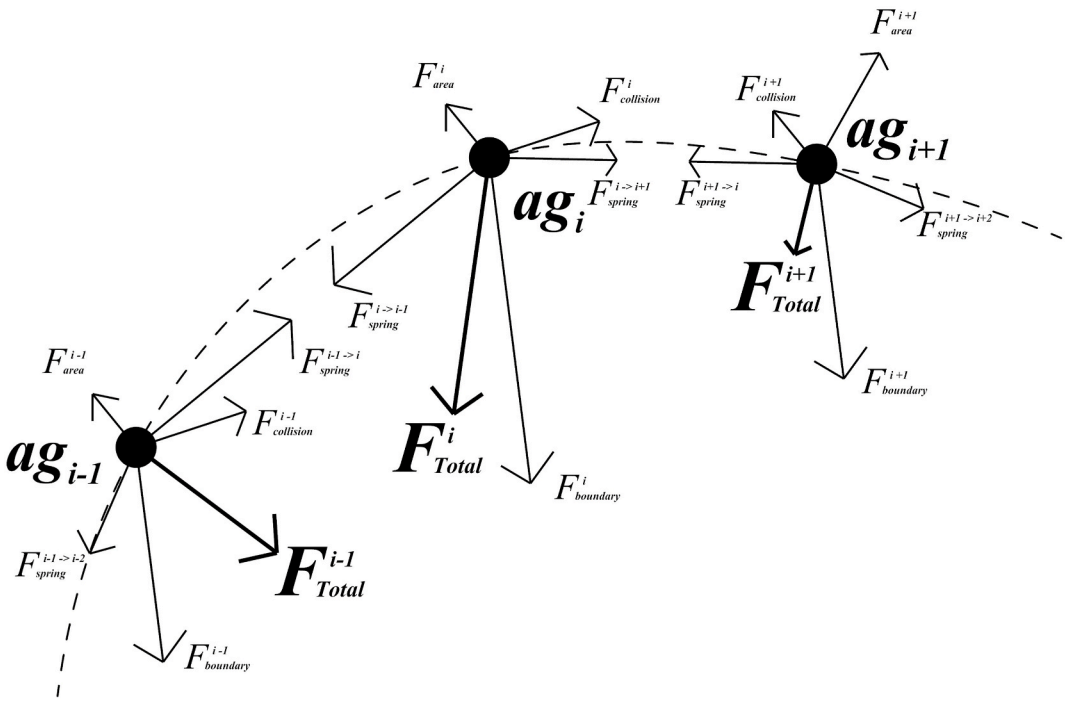

Fig. 7. Force vectors and the resultant force on each agent.

denoted as $F^i_{collision}$, thrusting them out of the other zone (Table 3). Fig. 8 depicts the force diagrams.

3.2. Agent-based modelling in the second hierarchical step

In the second hierarchical step, the sup-spaces of public and private zones denoted as S_i were randomly distributed inside the private and public boundaries that were resulted from the first hierarchical step. These sub-spaces were the ones included in the functional program. Each space was represented as a circle with a different radius, and the proportion of the circle's radiuses was directly related to the proportion of their space's desired area. The desired area of each space or the proportion of their area was an input parameter defined by the designer at the early stage. In contrast to the first hierarchical phase, the agent point in this stage was specified as the center of each circle. Each agent was moved repeatedly by the resulting force until it found a stable location inside the footprint (Fig. 9).

The experiment revealed that the combination of six force vectors, presented in Equation (2), could generate more accurate bubble diagrams based on pre-defined high-level constraints. $F^i_{springC}$ was defined to simulate the force vector between a space and a window. $F^i_{springT1}$ simulated the direct topological connections while $F^i_{springT2}$ simulated the indirect topological connections. $F^i_{collision}$ avoided the spaces collision. $F^i_{boundary}$ kept the circles inside the footprint, and $F^i_{magnetic}$ forced the agents to move to the centroid of their convex geometrical boundary and generated bubbles attracted to the centre of each space (Table 4).

$$F^i_{Total} = w_1 F^i_{springC} + w_2 F^i_{springT1} + w_3 F^i_{springT2} + w_4 F^i_{collision} + w_5 F^i_{boundary} + w_6 F^i_{magnetic} \qquad i \in s \tag{2}$$

Before explaining the details of each force, first, we should consider the high-level constraints precisely. The adjacency matrix and a sample building footprint are two main high-level constraints defined as the input to the algorithms. The designer specified each space's direct and indirect connectivity in a discrete numerical range in the adjacency matrix. In this case, if space required natural light, the connection between the space and the targeted window was weighted as number 1.5. If two areas had a direct topological link, they were weighted as number 1. If they had an indirect topological connection, they were weighted as number 0.5, and in case of no relation, they were weighted as number 0. These numbers defined the priority of topological connections, and they were applied in agent-based modeling. The building footprint was another high-level constraint specified by the designer (Fig. 10). The positions of the entry door and windows were marked in the footprint.

In Fig. 11, the second hierarchical step force vectors have been represented. The spring force $F^i_{springC}$ kept the space S_i near to the window W_j which guaranteed the topological connectivity of spaces and the building openings. The spring force $F^i_{springT}$ generated an attraction force between two areas with direct or indirect topology connectivity. The amount of the attraction force relied on the number associated with the link in the adjacency matrix. $F^i_{collision}$ was a repulsion force aiming to avoid collision between the spaces. $F^i_{boundary}$ was also a repulsion force that kept the spaces inside the footprint.

The resultant force F^i_{Total} was the overall combination of the forces which defined the movement direction of each agent. Each agent had different resultant force vectors in subsequent simulation iterations. Weight coefficients are non-zero positive factors that the designer can adjust to emphasize a particular force. Modifying these hyper-parameters could result in design alternatives while satisfying topological constraints. In this study, the weights are kept the same for all interacting agents. However, the designer could alter the weights for each agent to have more control over generating desired bubble diagram. Since several possible solutions are available for spatial layout design, this hyper-parameter helps the designers generate bubble diagram alternatives. Weight coefficients could also be utilized to develop zone arrangement possibilities in prior hierarchical phases.

3.3. Layout plan alternatives based on window constraints

Regarding the hierarchical approach of agent-based modeling and the defined forces, each agent moved iteratively until all agents reached an equilibrium phase, where there was no more active resultant force to move the agents. The topological constraint between a space and an opening was a key factor to generate several possible solutions. The user could help the algorithms to reach the equilibrium phase in fewer iterations by selecting the optimum windows for each space. Each selection could yield several possible solutions while satisfying the defined high-level constraints. Ignoring the openings would result in solutions meeting the topological constraints without any relation to the footprints' openings.

3.4. Convert bubble diagram to functional heat map

Following the above phases, and after hierarchical agent-based modeling, the algorithm created a bubble diagram in which the size

Table 3
Description of forces of the first hierarchical step.

Force	*Description*
F^i_{spring}	*The forces between adjacent agents*
F^i_{area}	*The forces to keep the area of the zone at a predefined value*
$F^i_{collision}$	*The forces to avoid collision between the two zones*
$F^i_{boundary}$	*The forces to keep the agents inside the exploration space*

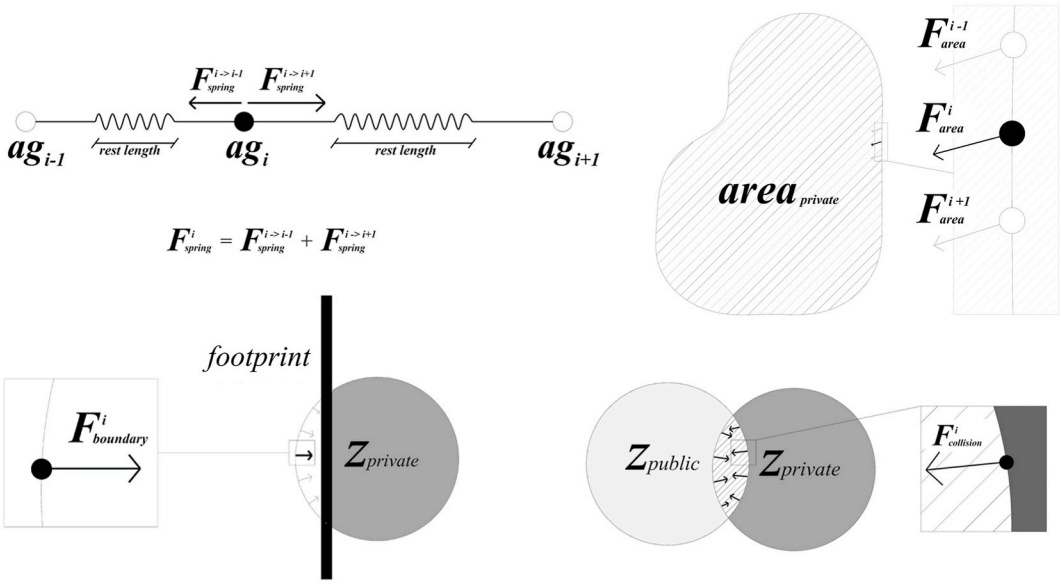

Fig. 8. Force diagrams of agents located on the perimeter of the zone's boundary line (first hierarchical loop).

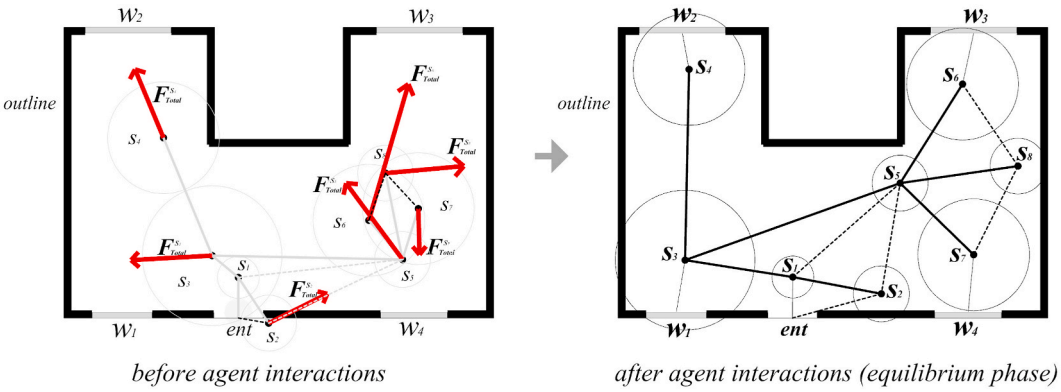

Fig. 9. Equilibrium phase of the second hierarchical step.

Table 4
Description of 2nd hierarchical step forces.

Force	*Description*
$F^i_{springC}$	*The attraction forces between the agents and the windows*
$F^i_{springT1}$	*The attraction forces between agents with direct topological connection*
$F^i_{springT2}$	*The attraction forces between agents with indirect topological connection*
$F^i_{collision}$	*The repulsion forces between the agents in case of collision*
$F^i_{boundary}$	*The backward forces on the agents in case of passing the search boundary*
$F^i_{magnetic}$	*The attraction forces on the agents to keep them near the centroid of the space*
w_j	*Weight coefficient hyperparameter (greater than zero)*

of each bubble was proportionate to its area, and the color of each circle reflected its function. Since the pix2pix model takes an image as input, converting the graph representation to an image visualization was critical. To achieve this, a conversion approach that provided a continuous representation while conveying the outcome of topological relations should have been used. Spatial heat maps vary continuously over space, and it was selected for the new representation of the topological relations. Thus, a rule-based algorithm was applied to turn the bubble diagram into a heat map. In doing so, the interior part of the apartment footprint was divided into small grid cells in the first step. In the next step, each grid was colored based on its proximity to the bubbles. Fig. 12 illustrates the final result.

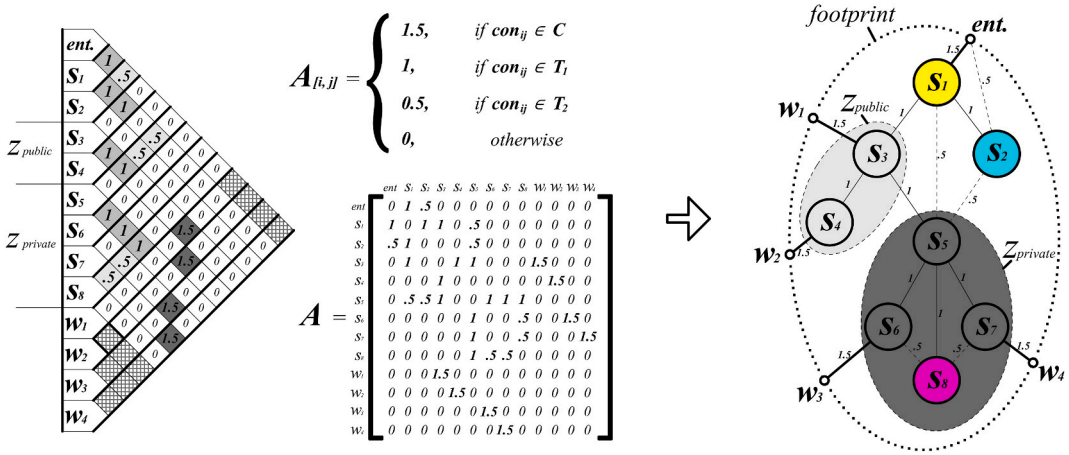

Fig. 10. Adjacency matrix & topological weighted graph.

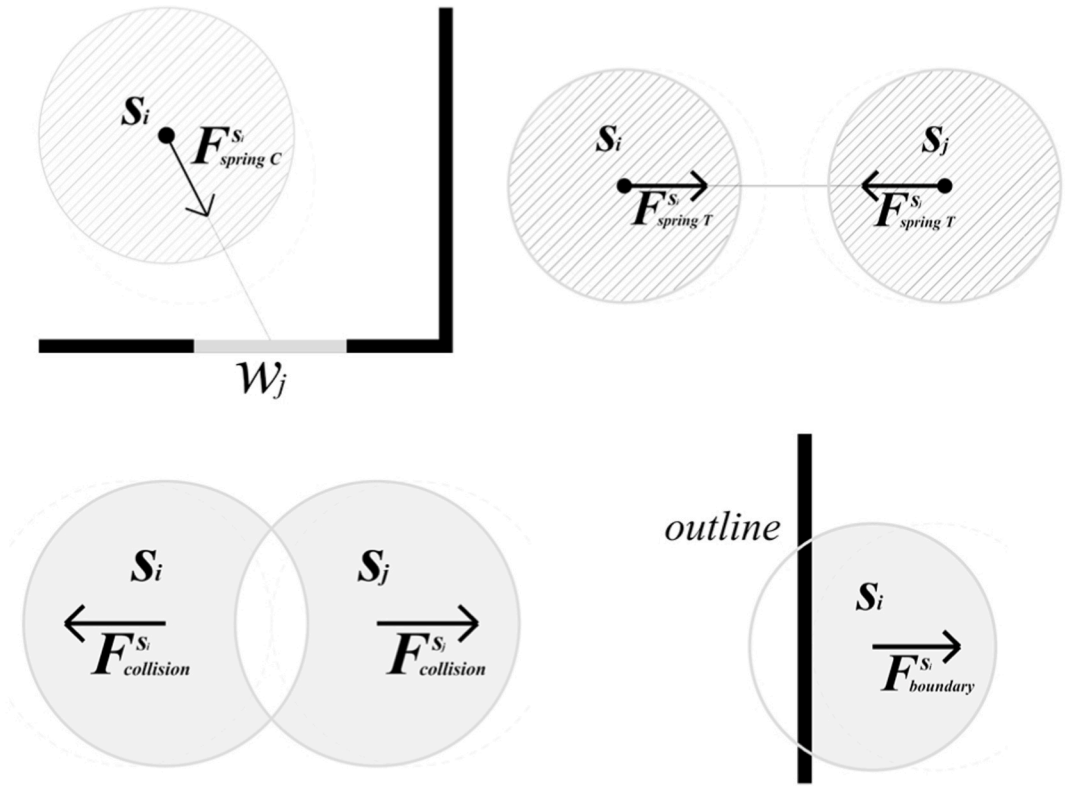

Fig. 11. Second hierarchical step force diagrams.

This heat map represented the probability of each grid cell's function in the given footprint. The resulting heat map looked smooth and continuous due to the small size of the grid cells. Each color in the heat map corresponds to a distinct function and is the same as the topological graph. Green is used for the kitchen, red for the bedrooms, yellow for the lobby and corridor, grey for the living room, cyan for the WC, and magenta for the bathroom. Different color palettes may be employed in the case of a complicated functional program.

4. Training and evaluation of cGAN

Over the last few years, numerous research works and algorithms have been presented, focusing on AI generative models. Their primary purpose was to generate synthetic data similar to real-world data. Some of these works have been successful in obtaining outstanding results. Auto-Encoders (AE), Variational Auto-Encoders (VAE) and Generative Adversarial Networks (GAN) are examples

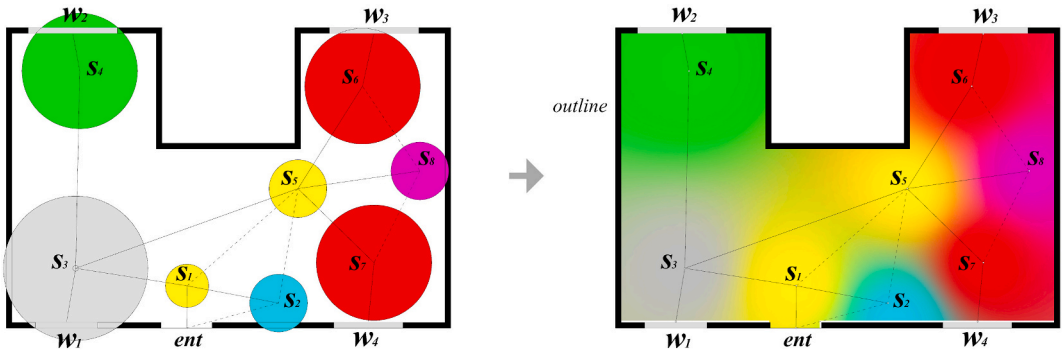

Fig. 12. Converting the bubble diagram to a functional heat map.

of AI generative models. GANs are the most recent developments, firstly introduced by Ian Goodfellow [14] in 2014, which led to a revolution in the field of AI generative design. Since 2014, many scholars have examined several variations of GANs for different purposes. cGAN is one of the GAN branches based on the input condition. In this study, pix2pix conditional GAN (cGAN) algorithm has been applied to generate synthetic images from input images. Pix2pix was introduced by Phillip Isola [13] in 2017, which has been one of the most significant algorithms in the field of generating synthetic images. Working with GANs consists of two steps: 1- Generate the training dataset 2- Train the model based on the training dataset.

4.1. Training dataset

The input dataset of this study included 320 pair images that the first one was a heat map and the second one was a layout design.

Fig. 13. A part of the training dataset (pairs of images).

The dataset was generated manually based on collected apartment plans by the authors of this paper. In cGAN, this dataset is called real data. The main task at this stage was to generate the dataset and train the cGAN algorithm. Conditional GAN accurately trains the mapping procedure from the first image to the second image. The cGAN was specifically trained on how to transform heat maps to layout designs.

In Fig. 13, some of the 320 dataset pairs of images are displayed. Each data contained a pair image of the input and the target known as ground truth. Colors in each data were low-level features representing architectural spaces. Red represented the bedroom, green the kitchen, yellow the entrance and hallway, grey the living room, cyan the WC and purple the bathroom. This part aimed to learn the geometrical constraints and rules from previous experiences. So the cGAN had to learn how to translate a heat map to a labelled layout with a correct proportion, area and boundary line. Satisfying geometrical constraints have always been the challenging part of automated space planning. Here, instead of defining the geometrical constraints as explicit rules, the dataset with geometrical semantics was shown to the cGAN. The algorithm implicitly learned how to generate a labelled layout from a heat map image. After training, the algorithm was ready to predict the new layouts according to the given heat maps.

4.2. Training the cGAN

The cGAN algorithm is based on the GAN algorithm, which consists of two adversarial networks named generator and discriminator, but they differ in terms of their input. In the GAN algorithm, the generator network starts from a noise vector. In an up-sampling process, a synthesized image is generated, which is then entered as the input of the discriminator network. The purpose of the discriminator network is to distinguish these fake images from real images provided in the dataset. When training the generator network, the goal is to optimize weights and biases to deceive the discriminator network. Training the discriminator network aims to optimize the discriminator's weights and biases to distinguish between fake and real images accurately. In subsequent iterations, the generator and discriminator are trained, which improves the capabilities of each network in a competitive challenge. That is why they are referred to as adversarial networks. After the training, the generator network can generate images similar to the real images provided in the dataset. It would be hard for the discriminator to distinguish it as fake data. Now, a reliable and robust generator is ready for generating synthetic data.

The cGAN is also based on the same concept of competing generator and discriminator networks. Unlike the GAN generator network, which starts with a noise vector, in cGAN, the generator starts from an input condition. In the pix2pix algorithm, this input condition is an image. Then the algorithm reduces the dimension of the input condition to a z-compressed latent space with a down-sampling process. After that, an up-sampling process returns the low-dimensional data to its original size. This output is the generated synthetic image. The pair images of the input condition and the generated image are shown to the discriminator as fake data. Meanwhile, the discriminator reads the pair of images of the input condition and the corresponding output image from the dataset as real data. The discriminator is trained based on these fake and real data. The generator is also trained in a way capable of deceiving the discriminator with its synthetic image. The trained discriminator classifies it as real data (based on the main concept of GAN). The heat map images were used as the input conditions, while the labelled layouts functioned as the ground truths. Fig. 14 illustrates the structure of the cGAN algorithm for translating heat maps to labelled layouts. In this process, the original heat-map image was transformed into a new image, which looked like a noise image in the first iterations. As the training proceeded, these noisy synthetic images changed to labelled layouts similar to the layouts provided in the dataset.

The weight and bias coefficients in generator and discriminator networks are optimized throughout the training phase. Each of these networks was trained separately; so that in each iteration, the discriminator network was first trained one step and then the generator network was trained one step. This operation was performed one by one during several iterations. During the training of the generator, the weight coefficients of the discriminator were constant. The important point in this training process is that both networks must be strengthened at the same time. In this case, the generator network, our ultimate goal, reached a high degree of accuracy. By the end of the training operation, the discriminator is no longer needed. We could generate the spatial location layout from the input data using the generator network, which in this study were heat-map images generated via agent-based modeling (Fig. 15).

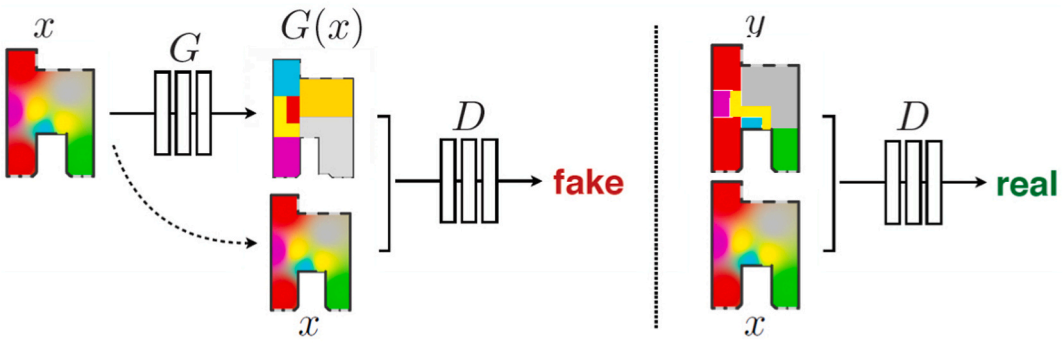

Fig. 14. Training the generator and discriminator in the cGAN.

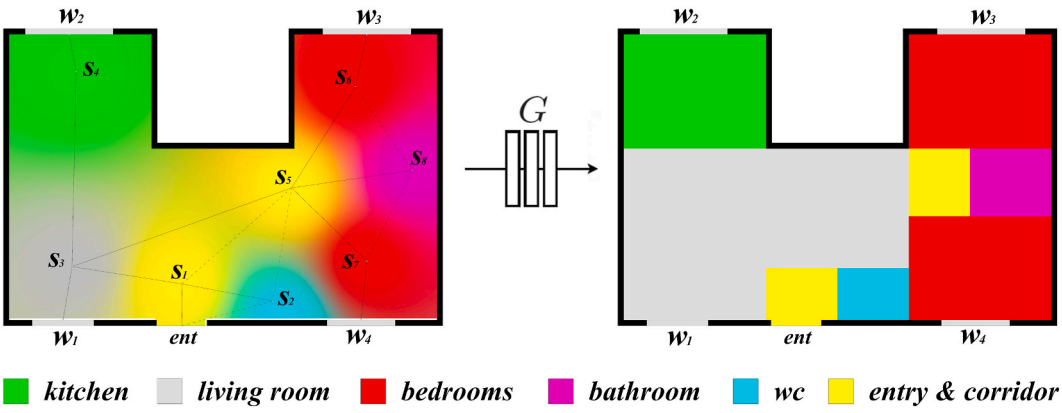

Fig. 15. Using trained generator for predicting layout design based on the heat map.

5. Results & discussion

Four new footprints were created as test cases to evaluate the efficiency of the agent-based modeling and cGAN algorithms (Fig. 16). The desired topological constraints were also defined (Fig. 17). The aim was to validate the accuracy of the proposed hybrid approach in generating layout design from the perspective of satisfying the topological and geometrical constraints.

Fig. 18 presents the results of the first hierarchical loop in agent-based modeling. At this step, the exploration space was partitioned into two zones. As defined in the adjacency matrix, the public zone included kitchen and living spaces which both of these spaces required direct topological connection with windows. Also, the private zone had two bedrooms which both required direct links with windows. Thus, it was expected from the first hierarchical step of the agent-based modeling to generate zones meeting these topological constraints. As illustrated in Fig. 18, it is evident that both zones in all test cases satisfy the high-level conditions, and they both have a direct connection with two openings. However, it seems that in test case number four, the public zone has a weak connection with its second window. But since these zones are used for randomly distributing the agents of the following hierarchical loop, it is expected to get stronger connections with the openings in the next loop.

In the second hierarchical loop, the agents were first distributed randomly in their zone's boundary. Then the interaction forced were defined as described in section 3. In Fig. 19, the final results of the second hierarchical loop of agent-based modeling are represented. It is the bubble diagram proposing the location of each space inside the footprint after reaching the equilibrium phase. The topological constraints should be examined to validate the outcomes of this section. In all four test cases, the kitchen, both bedrooms and the living room are located adjacent to an opening, and they all have a direct topological connection with windows. In all four cases, the lobby and WC are close to the entry door, meeting the requirements. The corridor and the bathroom are both adjacent to one other and close to the bedrooms. There are no other spaces between the living room and the kitchen; therefore, the relationship is topologically direct. The public zone of test case number four had a weak connection with two openings in the first hierarchical loop. Now it is completely compensated by the second loop simulation, and both sub-spaces of the public zone are located next to windows. All desired topological constraints are satisfied in all four test cases.

In the next part, these bubble diagrams were converted to a heat map to be loaded into a deep learning model. In the present study, more than 42,000 training steps were executed for the pix2pix algorithm. At the end of this process, the generator network was separated and used as a predictive algorithm. This generator received an image as input and generated an image as output. The first row in Fig. 20 represents the heat maps. These heat maps were loaded as inputs of the trained pix2pix algorithm and the second row are the generated layouts. As mentioned earlier, the deep learning algorithm was applied to satisfy geometrical constraints. All spaces in the generated layout were located at the same position where the heat map proposed earlier. Thus, the topological relationship was not changed and complied with the high-level constraints. The geometry of each space in all four test cases were generated based on the geometrical patterns learned from the training dataset. So, the dataset plays an influential role in defining geometrical constraints. Here, the designer could train the cGAN model with any prepared dataset to apply desired geometrical constraints to the results.

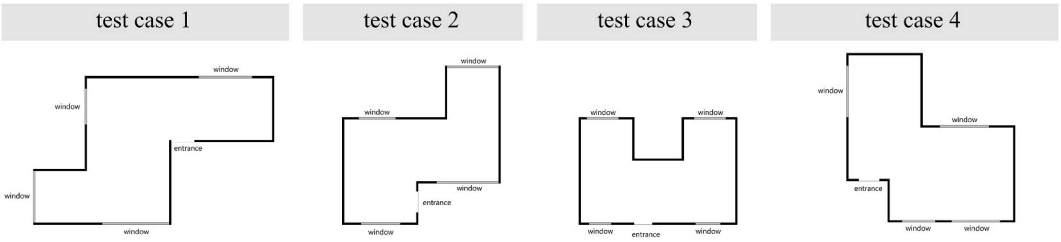

Fig. 16. Base footprints.

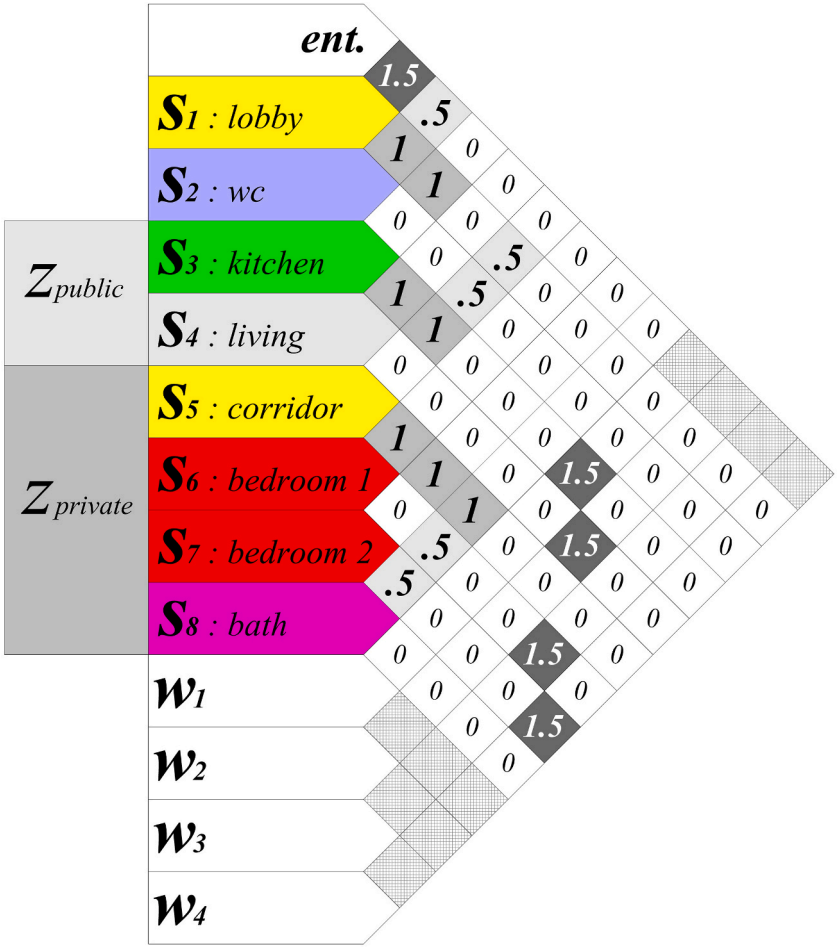

Fig. 17. Topological constraints specified by the designer.

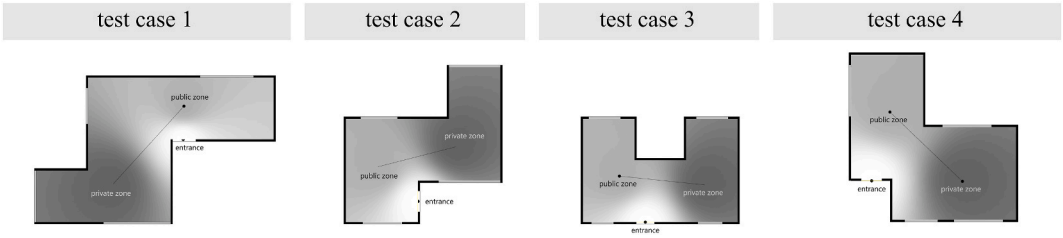

Fig. 18. Results of first hierarchical loop in agent-based modeling (private and public zones).

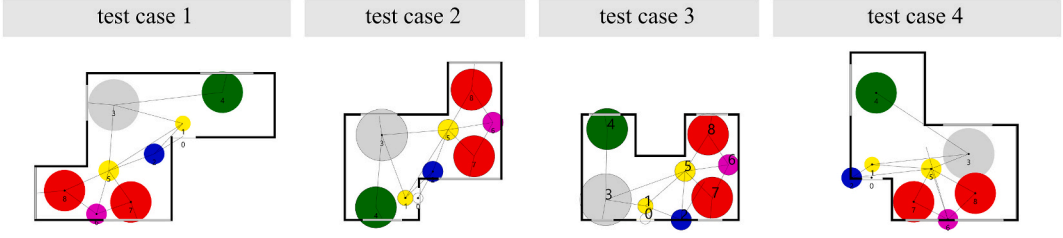

Fig. 19. Results of second hierarchical loop in agent-based modeling (bubble diagram).

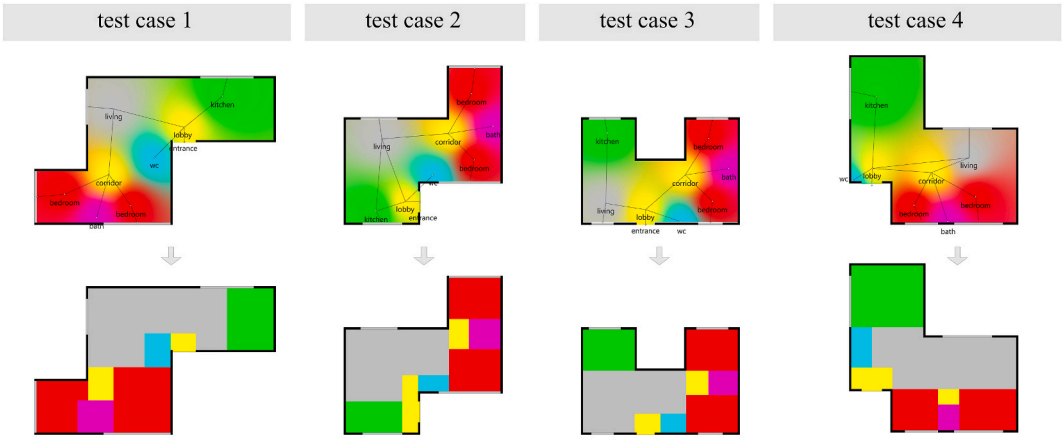

Fig. 20. Results of cGAN (generated layout design).

Regarding the orthogonal patterns of trained layouts, the generated spaces were also orthogonal. The area and the proportion of each space in the generated layouts were based on the learned patterns. For instance, in test case number one, the yellow color of the lobby at the entrance door occupied a large area in the heat map, whereas in the generated layout, it only occupied a reasonable rectangular space.

Comparing the results of this study to the previous studies reveals the power of the hybrid approach for generating more accurate layouts from the perspective of topological and geometrical constraints. In previous works that applied deep learning models for space planning, the deep learning model learned both topological and geometrical constraints. A large dataset is necessary to provide acceptable results with that method. Also, the designer has no control over the desired topological conditions. The hybrid approach presented in this study grants the designer a high level of control on the desired topological relationship while benefiting from deep learning capabilities to satisfy geometrical constraints. Fig. 21 represents the overall results of the four test cases.

As previously indicated, the resulting bubble diagrams and labaled layouts can be useful in the early phases of a design process. The proposed hybrid approach does not seek to replace designers; rather, it serves as a tool that employs computer power to solve space planning problems. Designers are still the final decision-makers, and they may choose the desired layout from the generated alternatives and post-process the results to present detailed architectural drawings.

6. Conclusion & future works

The focus of this research was to figure out how to create automated 2d layout designs that fulfilled both topological and geometrical requirements. The goal was to give the designers full control over the process of creating layouts based on specified topological restrictions and geometrical characteristics. A hybrid approach of agent-based modeling and deep learning was proposed to solve this problem. Agent-based modeling was used in a hierarchical procedure to generate desirable bubble diagrams in an apartment footprint. The hierarchical technique was used to get the agents simulated from a good starting position, and the outcomes were completely compatible with the desired topological relationship. The bubble diagram was then transformed into a heat map through a rule-based algorithm. Since it was impossible to satisfy geometrical restrictions with rules due to their complexity, a data-driven approach was proposed. A deep learning algorithm was used to translate the heat maps to labelled colored layouts that conformed to the desired geometrical constraints. The pix2pix algorithm, which is based on conditional GANs, was trained by the dataset of pair images (heat maps and layouts). The geometrical constraints were defined implicitly by the dataset, which was created specifically for this purpose. The results confirmed the hybrid approach's effectiveness in developing reliable and robust layouts for spatial location problems. More precisely, the main conclusions are as follows:

- The designer may control the process and run the algorithm to generate desired layouts by modifying the adjacency matrix, windows, entry door, footprint and weight coefficients. Alternatives to bubble diagrams may result from these adjustments. The designer only needs to define the base footprint and high-level constraints and fine-tune the hyperparameters to achieve layout design variations.
- The designer may also alter the geometrical characteristics by using the specially produced dataset to train the cGAN algorithm. This new dataset is only required if the designer desires specific geometrical features such as curved shapes.
- Since the topological constraints are satisfied through agent-based modeling, and the deep learning model is used to conform to geometrical features, the application of the hybrid approach could be extended to other case studies with different space programs and footprints.
- The concept of hierarchical agent-based modeling is not dependent on a dataset, so it can be generalized to other domains and architecture types. In healthcare architecture, for example, the primary bubble diagram of the design project could be created through several hierarchical steps. The architect would need to set the appropriate adjacency matrix and the footprint and openings to accomplish so. The algorithm generates the desired diagram. However, for generating the shape and geometry of spaces, a data-

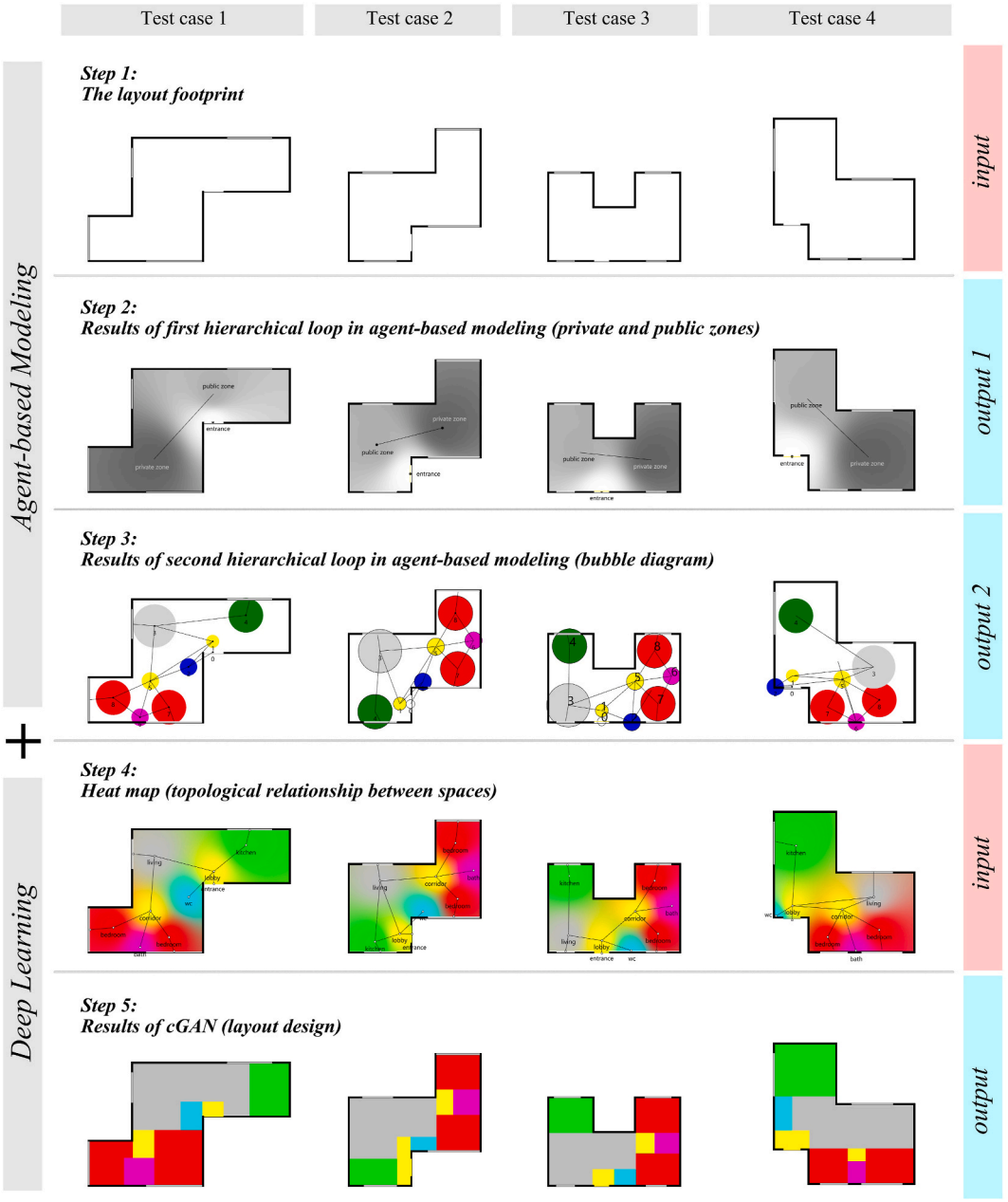

Fig. 21. Generated results through agent-based modeling and deep learning.

driven approach is applied. Since the geometrical features may differ in different architectural categories, preparing a specific training dataset with the desired geometrical features is crucial.

- Compared to previous works, the hybrid approach separates the topological and geometrical domains. In previous works that applied deep learning for automated layout design, the proposed model was responsible for satisfying both topological and geometrical constraints. That approach had two main drawbacks. First, they required large data set, and in the absence of such training data, it generated poor quality results mainly in terms of geometrical features. Second, the designer had no control over conditioning the algorithm to a desired adjacency topology. This paper's proposed hybrid approach overcomes these drawbacks by setting apart the two fields of topology and geometry. In this approach, the deep learning model is only responsible for geometrical attributes. Topological constraints are addressed by the agent-based model, which may be conditioned to meet desired high-level criteria. Also, conditioning the cGAN model to a heat map converted from a bubble diagram with correctly allocated spaces made it an easier task for the deep learning model to generate better layouts.

This approach can be developed in future studies to predict 3D layouts of multi-story buildings with topological relationships in

three-dimensional exploration space. The same hierarchical agent-based modeling concept could be used to generate 3D zones and spaces. In the final hierarchical step, the agents will be randomly distributed for initiating the simulation. A 3D graph with spheres will satisfy the topological requirements in the defined 3D boundary. From the perspective of geometrical constraints, a new three-dimensional representation should be devised for preparing the training dataset.

CRediT authorship contribution statement

Morteza Rahbar: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing – original draft, Writing – review & editing, Data curation, Visualization. **Mohammadjavad Mahdavinejad:** Supervision, Project administration, Formal analysis. **Amirhossein Davaie-Markazi:** Supervision, Project administration, Formal analysis. **Mohammadreza Bemanian:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M.I. Campbell, R. Koenig, K. Knecht, Comparing two evolutionary algorithm based methods for layout generation: dense packing versus subdivision, AI EDAM (Artif. Intell. Eng. Des. Anal. Manuf.) 28 (2014) 285–299.
[2] I.G. Dino, An evolutionary approach for 3d architectural space layout design exploration, Autom. ConStruct. 69 (2016) 131–150.
[3] J.H. Jo, J.S. Gero, Space layout planning using an evolutionary approach, Artif. Intell. Eng. 12 (1998) 149–162.
[4] P.H. Levin, Use of graphs to decide the optimum layout of buildings, Architects' J. 7 (1964) 809–815.
[5] J. Grason, A Dual Linear Graph Representation for Space-Filling Location Problems of the Floor Plan Type, Emerging Methods in Environmental Design and Planning, 1970.
[6] J. Grason, An approach to computerized space planning using graph theory, in: Proceedings of the 8th Design Automation Workshop, pp. 170–178 .
[7] R.S. Liggett, W.J. Mitchell, Optimal space planning in practice, Comput. Aided Des. 13 (1981) 277–288.
[8] J.S. Gero, V.A. Kazakov, Learning and re-using information in space layout planning problems using genetic engineering, Artif. Intell. Eng. 11 (1997) 329–334.
[9] R. Jagielski, J.S. Gero, A genetic programming approach to the space layout planning problem, in: CAAD Futures 1997, Springer, 1997, pp. 875–884.
[10] J.S. Gero, V.A. Kazakov, Evolving design genes in space layout planning problems, Artif. Intell. Eng. 12 (1998) 163–176.
[11] J. Michalek, R. Choudhary, P. Papalambros, Architectural layout design optimization, Eng. Optim. 34 (2002) 461–484.
[12] R. Baušys, I. Pankrašovaite, Optimization of architectural layout by the improved genetic algorithm, J. Civ. Eng. Manag. 11 (2005) 13–21.
[13] P. Isola, J.-Y. Zhu, T. Zhou, A. A. Efros, Image-to-image translation with conditional adversarial networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1125–1134 .
[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Adv. Neural Inf. Process. Syst. 27 (2014) 2672–2680.
[15] M. Mirza, S. Osindero, Conditional Generative Adversarial Nets, arXiv preprint arXiv:1411.1784, 2014.
[16] H. Rittle, M.M. Webber, Planning Problems Are Wicked, 1973. http://cec.prodwebb.lu.se/sites/cec.prodwebb.lu.se/files/rittel_and_webber_1973_planning_problems_are_wicked_problems.pdf. Retrieved from:.
[17] H.A. Simon, The Sciences of the Artificial, MIT press, 2019.
[18] R. Koenig, S. Schneider, Hierarchical structuring of layout problems in an interactive evolutionary layout system, AI EDAM (Artif. Intell. Eng. Des. Anal. Manuf.) 26 (2012) 129–142.
[19] J. Roth, R. Hashimshony, A. Wachman, Turning a graph into a rectangular floor plan, Build. Environ. 17 (1982) 163–173.
[20] J. Roth, R. Hashimshony, Algorithms in graph theory and their use for solving problems in architectural design, Comput. Aided Des. 20 (1988) 373–381.
[21] K. Shekhawat, Automated space allocation using mathematical techniques, Ain Shams Eng. J. 6 (2015) 795–802.
[22] K. Shekhawat, Enumerating generic rectangular floor plans, Autom. ConStruct. 92 (2018) 151–165.
[23] G. Ślusarczyk, Graph-based representation of design properties in creating building floorplans, Comput. Aided Des. 95 (2018) 24–39.
[24] X.-Y. Wang, Y. Yang, K. Zhang, Customization and generation of floor plans based on graph transformations, Autom. ConStruct. 94 (2018) 405–416.
[25] S. Arifin, R. Putri, Y. Hartono, E. Susanti, et al., Developing ill-defined problem-solving for the context of "south sumatera", in: Journal of Physics Conference Series, volume vol. 943, p. 012038 .
[26] U. Flemming, et al., A Generative Expert System for the Design of Building Layouts: Version 2, 1988. Technical Report.
[27] J.P. Duarte, A discursive grammar for customizing mass housing: the case of siza's houses at malagueira, Autom. ConStruct. 14 (2005) 265–275.
[28] G. Stiny, W.J. Mitchell, The palladian grammar, Environ. Plann. Plann. Des. 5 (1978) 5–18.
[29] S.-P. Li, J.H. Frazer, M.-X. Tang, A constraint based generative system for floor layouts, in: Proceedings of the Fifth Conference on Computer Aided Architectural Design Research in Asia, CUMINCAD, 2000.
[30] Z. Guo, B. Li, Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system, Front. Archit. Res. 6 (2017) 53–62.
[31] X. Bonnaire, M.-C. Riff, A self-adaptable distributed evolutionary algorithm to tackle space planning problems, in: International Workshop on Applied Parallel Computing, Springer, pp. 403–410 .
[32] S.S. Wong, K.C. Chan, Evoarch: an evolutionary algorithm for architectural layout design, Comput. Aided Des. 41 (2009) 649–667.
[33] M. Inoue, H. Takagi, Layout algorithm for an ec-based room layout planning support system, in: 2008 IEEE Conference on Soft Computing in Industrial Applications, IEEE, pp. 165–170 .
[34] M. Inoue, H. Takagi, Emo-based architectural room floor planning, in: 2009 IEEE International Conference on Systems, Man and Cybernetics, IEEE, pp. 518–523 .
[35] E. Rodrigues, A.R. Gaspar, Á. Gomes, An approach to the multi-level space allocation problem in architecture using a hybrid evolutionary technique, Autom. ConStruct. 35 (2013) 482–498.
[36] T. Wortmann, A. Costa, G. Nannicini, T. Schroepfer, Advantages of surrogate models for architectural design optimization, artificial intelligence for engineering design, analysis and manufacturing, AI EDAM (Artif. Intell. Eng. Des. Anal. Manuf.) 29 (2015) 471.
[37] M. Rahbar, M. Mahdavinejad, M. Bemanian, A.H. Davaie Markazi, L. Hovestadt, Generating synthetic space allocation probability layouts based on trained conditional-gans, Appl. Artif. Intell. 33 (2019) 689–705.
[38] W. Huang, H. Zheng, Architectural Drawings Recognition and Generation through Machine Learning, 2018.
[39] S. Chaillou, Archigan: artificial intelligence x architecture, in: Architectural Intelligence, Springer, 2020, pp. 117–127.